

Intelligent tutors for all: Constraint-based modeling methodology, systems and authoring

Contributing Authors

Antonija Mitrovic, Brent Martin, Pramuditha Suraweera
Intelligent Computer Tutoring Group
University of Canterbury, Christchurch, New Zealand

Abstract: Intelligent Tutoring Systems (ITS) have revolutionized online education by providing individualized instruction tailored towards each student. Constraint-based tutors model instructional domains at an abstract level, a novel approach that simplifies the development of ITSs. We have developed many effective constraint-based tutors over the last decade in a number of instructional domains of various characteristics, some of which have been successfully commercialized. Constraint-based tutoring is now a mature and successful approach to providing adaptive learning environments. Our authoring tools aim to make this technology widely available to teachers and students everywhere.

Keywords: **Artificial Intelligence:** Applications and Expert Knowledge-Intensive Systems: Education, Knowledge Representation Formalisms and Methods, Knowledge Modeling; **Computers and Education:** Computer Uses in Education: Computer-assisted instruction

Introduction

Providing Web-based environments for learning is increasingly important in today's society as the number of people with Internet access is growing and the resources for supporting effective learning are restricted. The idea of using computers in education is not new. There are numerous e-learning systems available; however, most of them are overly simple and provide very limited interactivity to their users. For learning to be effective, the student must be active and have opportunities to practice important skills. Intelligent Tutoring Systems (ITS) provide tailored support to each individual student, a unique strength that comes from their ability to model various types of knowledge required for instruction: domain knowledge, knowledge about their students (i.e. student models), models of pedagogy and communication knowledge.

Since the inception of ITS more than three decades ago many approaches to developing them have been proposed, most of which have stayed in the realm of research labs. The same is true for the actual tutors developed – typically systems were only used under very strict experimental conditions in research labs, often with participants who are paid for their time and not representative of typical students. However, there are some ITSs that have been widely used in schools, such as model tracing tutors¹, developed by researchers at Carnegie-Mellon University.

In the last decade a promising new type of ITSs, referred to as Constraint-based Tutors, has emerged. We at the Intelligent Computer Tutoring Group (ICTG) have developed constraint-based tutors for instructional domains of very different natures. Our approach to building ITSs is based on Stellan Ohlsson's theory of learning from performance errors², which resulted in the methodology known as constraint-based modeling³ (CBM). A distinguishing characteristic of constraint-based tutors is the knowledge representation formalism they are based upon. Ohlsson proposed that knowledge should be represented in the form of constraints, which specify what ought to be so, rather than generating problem-solving paths. Domain knowledge is thus used as a way of prescribing abstract features of correct solutions, rather than as a recipe for performing tasks in a domain, the way it is done in model tracing (using production rules). Constraints support evaluation and judgment, not inference, and are used to represent both domain and student knowledge.

This paper follows the evolution of CBM from an abstract idea to a methodology proven through the development of effective tutoring systems and powerful ITS authoring tools. We start by discussing constraints as a knowledge-representation formalism, and present features of some example systems. Our constraint-based systems have been evaluated thoroughly in real classrooms with students of various backgrounds. The results show that CBM is a sound theoretical foundation for ITSs, and demonstrate that our development methodology is effective and efficient. We present details of some evaluation studies, and describe commercial success.

The effort required to build an ITS is a major impediment to their wide-spread adoption⁴. ICTG has developed constraint-based authoring tools that dramatically reduce this effort. We describe WETAS, a tutoring shell, and ASPIRE, a comprehensive authoring platform, and discuss how they aim to meet the challenge of enabling educators to build tutors as teaching tools for their own courses.

Constraint-based tutoring is now a mature and successful approach to providing adaptive learning environments. Experimental results show that our approach is equally effective in supporting student learning as state-of-the-art ITSs today, and at the same time requires less effort to be developed⁵. However, the benefits of CBM still have not been fully explored. We are currently extending CBM to model meta-cognitive skills, such as self-explanation and collaborative skills. We describe these future directions and present our vision of how CBM may lead to widespread access to technology that enhances learning for all.

Representing Knowledge as Constraints

Constraint-based modeling uses abstraction to avoid the need to model students' misconceptions. Constraints represent only correct knowledge in terms of pedagogically significant states; each constraint maps to a set of solution states that share the same domain principle. A constraint-based ITS can therefore react in the same way (e.g. by displaying the same feedback message) for any solution that violates a given constraint. Constraints have three components: a *relevance* condition, a *satisfaction* condition and

the feedback message. The relevance condition describes (in terms of problem/solution features) when this constraint is applicable. The satisfaction condition then specifies additional tests to be applied to the solution to check its correctness. This way, constraint violations allow an ITS to react at the right time, and also govern the instruction to be delivered. The feedback message attached to the constraint tells the student that his/her solution is wrong, points out why it is wrong, and reminds the student of the corresponding declarative knowledge (i.e. the domain principle that is violated by the solution). An example constraint is *“If you are driving in New Zealand, you need to be on the left side of the road.”* The relevance condition of this constraint indicates it is applicable to those driving events that take place in New Zealand. For such an event to be valid, it also needs to meet the satisfaction condition (i.e. the car is on the left). If this latter condition is not met, appropriate feedback can be given, such as *“When driving in New Zealand you need to keep left”*. An interesting side-effect of this approach is that CBM is silent to situations it has no knowledge of. We describe therefore CBM’s approach as “innocent until proven guilty”: if there are no constraints violated by a particular solution, it is deemed correct. This is in contrast to approaches that enumerate possible problem-solving paths such as model tracing; if a student performs an action that is not on a known path, a model tracing tutor will typically assume the answer is incorrect (although it might not know why); otherwise it would be unable to follow what the student is doing. Both approaches have their strengths and weaknesses, but it means that overall CBM is more *permissive* than model tracing.

Just as constraints are used to represent the domain knowledge by specifying features of correct solutions in the domain, they also serve as the basis for representing *student’s* knowledge. When a student submits a solution, a constraint-based tutor analyses it using the constraints; relevant constraints are identified, and their satisfaction conditions determine whether they have been satisfied or violated. The lists of relevant, satisfied and violated constraints thus serve as a short-term student model, which is then used to update the long-term model of the student’s understanding. A student’s knowledge may be represented in constraint-based tutors in many ways, such as an overlay on top of the domain model, as a set of performance histories for all constraints used by the student, or even a Bayesian student model.

Pedagogically CBM determines the content of instruction provided. If there are errors in a student’s action, the ITS will present feedback provided by the violated constraints. The form of this feedback is shaped by the underlying learning theory: it should tell the student what domain principle he/she has violated, how it was violated by the student’s solution, and reiterate the correct domain principle². However, the style and delivery of feedback is independent of CBM; they can be adapted to a particular student. For example, feedback can be given in textual and/or pictorial form, depending on the student’s learning style. Further, both the timing (immediate or delayed) and amount of feedback (i.e. the number of feedback messages and the level of detail) can vary and can be adaptive.

The student model is also used for problem selection, by directing the ITS to select problems that exercise constraints the student has yet to master. Many problem-solving

strategies can be implemented, and CBM merely supports the process by providing information about the mastery and novelty of concepts and exercises.

CBM is computationally simple, requiring only pattern matching. Unlike model tracing, it does not require a runnable expert module; student diagnosis is achieved by using the constraints to compare the student's solution to a pre-specified "ideal" solution. This works even in domains where problems can have multiple (and often radically different) correct solutions because constraints can identify alternative constructs in the student solution that are equally valid⁶. The ideal solution therefore simply encapsulates the semantics required of the student's answer. If a problem solver is available, however, it can be used beneficially to provide more information to the student in the form of advice about the next step to perform. Alternatively, the constraints themselves can be used to repair errors in the student's answer, and the repaired solution (or part of it) can then be shown to them as a hint of what to do/fix next.

CBM has further advantages over other modeling techniques. It does not require bug libraries (collections of common errors students make), which are difficult and expensive to collect and maintain. Constraint-based tutors are also robust in the face of student inconsistency; CBM does not model problem-solving procedures, and therefore it can handle mixed problem-solving strategies. This also makes CBM able to handle creativity because the student is free to use a novel problem-solving procedure if they wish without the system intervening. However, the author may optionally add constraints that catch suboptimal strategies, in which case the student will be guided towards optimal behaviour. CBM's higher level of abstraction also means that domain models are smaller, and therefore authoring effort is reduced⁵.

Successful Constraint-based Tutors

SQL-Tutor⁶ is the first constraint-based tutor built by the ICTG. It supports students learning how to query relational databases using SQL. This language is hard for students to learn. It is typically taught in lectures and labs, with students learning query definition in the context of a specific database management system (DBMS). There is a great deal of complexity involved: students need to learn the concepts of the relational data model, be familiar with the DBMS used, and they also need to learn the syntax of SQL. On top of all of these difficulties, DBMSs provide error messages that are cryptic, hard to understand and limited to syntax errors only.

SQL-Tutor provides rich support to students, both in terms of the scaffolding provided by its interface and in terms of adaptive problem-solving support. The problem-solving interface (Fig. 1) provides problem text, the solution structure and information about the database schema. The student can also run their queries and see the resulting data. The system provides feedback on demand at several levels of detail. For the first submission, the student is only told whether or not the solution is correct. If they continue to submit incorrect solutions, they will progressively get more help until they are able to complete the problem or ask for the solution. The student model is used to select the next problem

to be posed to the student, and we have experimented with a number of different problem-selection strategies.

SQL-Tutor contains almost 700 constraints describing the fundamental principles that all solutions must satisfy. Some constraints check that the student is using correct syntax. For example, constraint 254 first checks whether the student has specified a nested SELECT statement in the WHERE clause (relevance condition); if that is the case, the satisfaction condition requires that the nested query is preceded by either a comparison operator, or a predicate (IN, ALL, ANY or EXISTS). Other constraints check whether the student's solution is the correct solution for the given problem by comparing it to the ideal solution. Such constraints (which we refer to as semantic constraints) also check for alternative ways of solving problems. As an illustration, constraint 263 is relevant for

The screenshot shows the SQL-Tutor interface with four main components labeled A, B, C, and D:

- A (Problem Text):** "List the numbers and titles of all movies made between 1990 and 1993."
- B (Workspace):** A query editor with the following content:


```
SELECT number, title
FROM movie
WHERE year >= 1990
```
- C (Feedback Panel):**
 - Message: "Almost there - you made 2 mistakes. Check that you have defined all search conditions needed in WHERE!"
 - Message: "You can correct your query and press 'Submit' again, or try getting some more feedback. Would you like to have another go?"
- D (Database Schema):**

The general description of the database is available [here](#). Clicking on the name of a table brings up the table details. Primary keys in the attribute list are underlined, foreign keys are in *italics*.

Table Name	Attribute List
<u>DIRECTOR</u>	<u>number</u> lname fname born died
<u>MOVIE</u>	<u>number</u> title type aanom aawon year critics <i>director</i>
<u>STAR</u>	lname fname <u>number</u> born died city
<u>CUSTOMER</u>	lname fname <u>number</u> address rentals bonus jdate
<u>TAPE</u>	<u>code</u> <i>movie</i> pdate times <i>customer</i> hiredate
<u>STARS IN</u>	<i>movie</i> <u>star</u> role

Figure 1. The interface of SQL-Tutor consists of (a) problem text, (b) workspace for composing SQL queries, (c) feedback panel and (d) database schema.

situations when the ideal solution contains a BETWEEN condition in the WHERE clause using an attribute, and the student has not used that predicate, but instead has a condition comparing the same attribute to a numerical constant. The satisfaction condition of this constraint requires that the student has also specified another (conjunctive) condition comparing the same attribute to another numeric constraint, thus allowing an alternative way of specifying a range of values for the attribute. Please note that this constraint only requires two search conditions in the student's solution using the same attribute (which is necessary for a range check), and does not check whether the student has used the correct

constants and comparison operators. These additional checks will be performed by other constraints, thus allowing feedback to be very precise. Figure 1 shows feedback given when constraint 263 is violated.

The success of SQL-Tutor led us to research the generality of CBM by implementing ITSs in other domains, and also to explore how the methodology could be enhanced. We have particularly focused on design tasks because these tasks are typically difficult to support using other types of ITSs. Since developing SQL-Tutor, we have also built EER-Tutor (the early, stand-alone version of which was KERMIT⁷), a tutor that teaches conceptual database design. This task is difficult for students because as well as requiring the student to learn how to construct EER diagrams, it also involves analytical experience that can only be obtained by practice. EER-Tutor provides numerous problems to students, a custom drawing tool for developing diagrams, and feedback at several levels of detail. The interface (Figure 2) reminds the student of the requirements for the current problem and provides drawing tools corresponding to the constructs of Enhanced Entity-Relationships (EER). It also reinforces good practice in the domain by asking the student to name the diagram components by selecting terms from the problem description. Although this latter mechanism is arguably somewhat restrictive, it is valuable from a pedagogical point of view in that it focuses the student on the problem requirements and forces him/her to use the end user's language; this is widely considered good practice in software engineering. The system also highlights the selected names (within the problem text) in various colors, thus providing a means for the student to visually explore how much of the problem he/she has covered. If there are any errors in the student's solution, the corresponding parts of the diagram will be highlighted in red, to help the student identify the errors in combination with the feedback given.

The majority of our tutors are developed for the area of database systems because we teach these areas and desired the tutors for our own courses. However, we have also developed ITSs in other domains to demonstrate that our approach is general. CAPIT and LBITS are systems that teach various aspects of the English language to elementary school children; the former teaches punctuation and capitalization rules, while LBITS contains a number of vocabulary building activities such as unscrambling words, turning the singular form of nouns into plural, and turning verbs into nouns.

All of the tutors described thus far were developed for domains that do not prescribe a strict problem solving procedure. We have also explored the suitability of CBM for procedural tasks, for which there are problem-solving algorithms available and therefore problem solvers can be developed. The first such system was NORMIT, a tutor that teaches data normalization. Data normalization is the process of refining a database schema by applying a series of tests to its relations and decomposing them if necessary. Although the algorithm is straightforward, students often fail to perform it correctly because they lack a strong understanding of the underlying database theory. NORMIT strengthens students' performance in applying the procedure by focusing on each step of the algorithm in isolation, and requiring them to complete the current step before being allowed to move to the next one. At each step NORMIT provides a summary of the

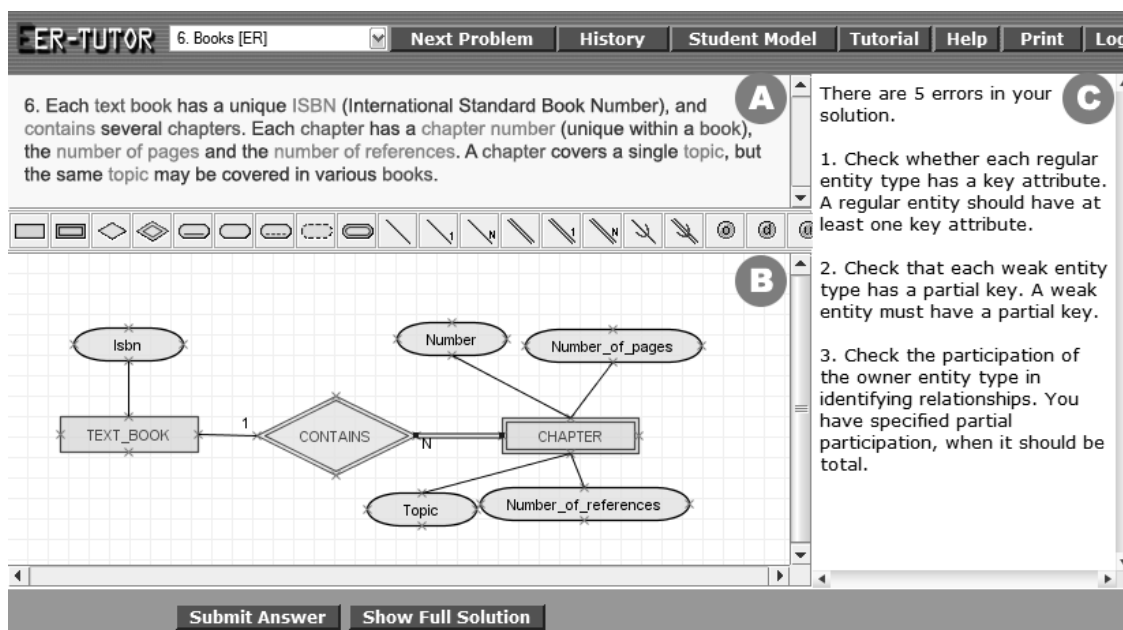


Figure 2. EER-Tutor interface containing (a) problem description, (b) EER diagram workspace, and (c) feedback for three violated constraints.

solution developed so far, and checks whether the student has completed all the necessary actions using the constraints relevant for that step. We experienced no difficulties generating constraints to support learning in this domain, and have since developed tutors for other procedural domains.

Evaluating Learning Effectiveness

Our firm belief is that ITSs must be evaluated in authentic classroom situations. We have performed more than 30 evaluation studies, all with students taking appropriate courses. SQL-Tutor alone has been evaluated in eleven studies at the University of Canterbury since 1998, focusing on a variety of issues such as learning gain, effectiveness of feedback, problem/feedback selection strategies, open student models and students' self-

assessment skills. Although classroom studies are much more difficult to organize and conduct than strictly controlled laboratory studies with paid participants, they are sorely needed because they provide information about representative student populations.

In the evaluation studies performed, we typically begin by measuring the students' existing knowledge via pre-tests. Participants are free to use the tutor under study as much as they wish over a certain fixed period of time (usually several weeks). A typical study involves two or more versions of the same system, which allows us to compare the effectiveness of various kinds of support implemented. The tutoring system collects data about all actions the students perform and stores it in log files. Students are typically asked to sit a post-test at the end of interaction, and are sometimes also asked to provide their impressions of the system. This information is then analyzed statistically in a variety of ways. When measuring the learning effectiveness of students using our systems, typical effect sizes we obtain are in the order of one standard deviation improvement, as good as that achieved by state-of-the-art ITSs available today.

SQL-Tutor was the first of our systems to be formally evaluated, with the results demonstrating that students using it achieved real learning gains. After only two hours with SQL-Tutor, students outperformed their peers in the final examination, scoring an average of three quarters of a standard deviation higher on questions related to SQL query formulation⁶. This result is comparable to results achieved by other approaches^{8,9}, including studies where the students have used the relevant system throughout an entire semester.

EER-Tutor has also been proven to be effective. The stand-alone version (KERMIT) was evaluated in August 2001 during a regular lab session at the University of Canterbury⁷. The 62 volunteers that participated in the study were randomly assigned to use the complete version of KERMIT (the experimental group) or a cut-down version of KERMIT that only provided the final solution (the control group). Pre/post test results revealed that the students who used KERMIT attained significantly higher gains ($t = 3.07$, $p < 0.01$). The effect size of the experiment, which allows the comparison of the results of one pedagogical experiment to another, was 0.63. The power of the experiment was 0.75 at significance 0.05, indicating that there is a high probability that the experiment would produce significant results for the same design, the same number of participants and the same effect size.

Commercial Success: The Database Place Web Portal

Having demonstrated the efficacy of constraint-based tutors at our own institution, the next step was to unleash them internationally, which we achieved through commercialization. Since February 2003, two of our database tutors (SQL-Tutor and NORMIT) have been available on the Addison-Wesley's "Database Place" Web portal (www.aw-bc.com/databaseplace), with EER-Tutor following a year later. The student population on Database Place is very different from our local students – anyone who has bought database textbooks published by Addison-Wesley may access this site. Unlike our domestic students, we know nothing about these students' backgrounds and pre-existing

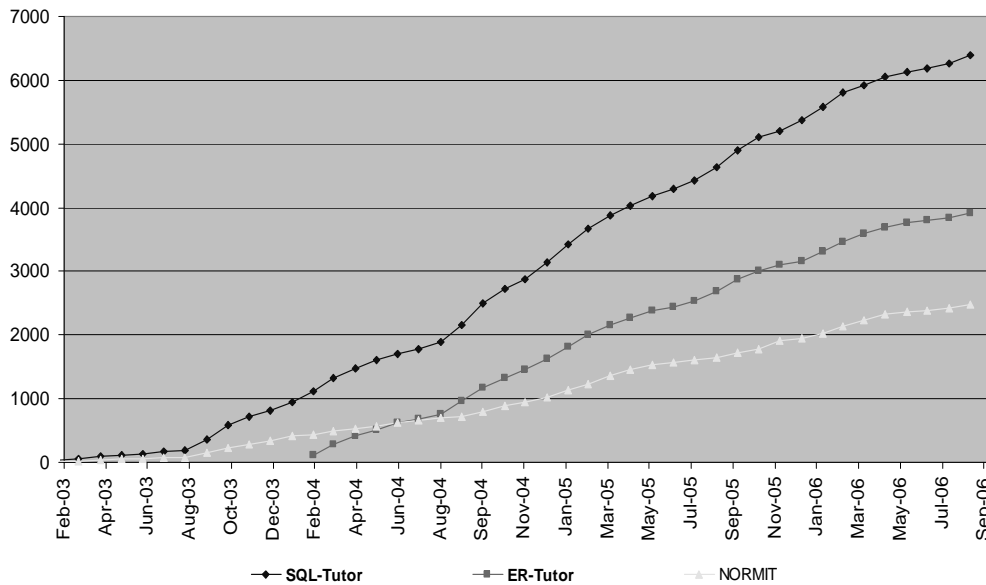


Figure 3. Registered Database Place Users

knowledge, so it is very interesting to observe the way they use the systems and their learning performance compared to our local cohort.

Figure 3 shows that the number of registered users for the three systems on Database Place increases almost linearly. There are periods of slower increase during northern hemisphere summers, which provides some evidence of the locations of the majority of users. Our contract with Addison-Wesley prevents us from requiring pre/post test to be completed by users, and therefore we cannot compare the test results of our Canterbury students to Database Place users. However, we log all students' actions both at Canterbury and at Database Place (including what constraints were satisfied and violated for each student submission), and can therefore compare the two populations in terms of their learning processes. Figure 4 shows the learning curves for the Canterbury and Database Place student populations. A learning curve plots the performance on using constraints in terms of the probability on average of violating a constraint versus the number of times the constraint has been applied. Psychological research has shown that such performance improves approximately as a power law of practice, and our tutors exhibit this characteristic. As can be seen in Figure 4, the two groups of students learn the domain material in comparable ways; the initial probabilities of errors (the y-intercepts) and the learning rates (the exponent of the power law functions) are similar. The degree of fit to a power law is a little higher for the Database Place population, but this is probably because the data volume is several orders of magnitude greater.

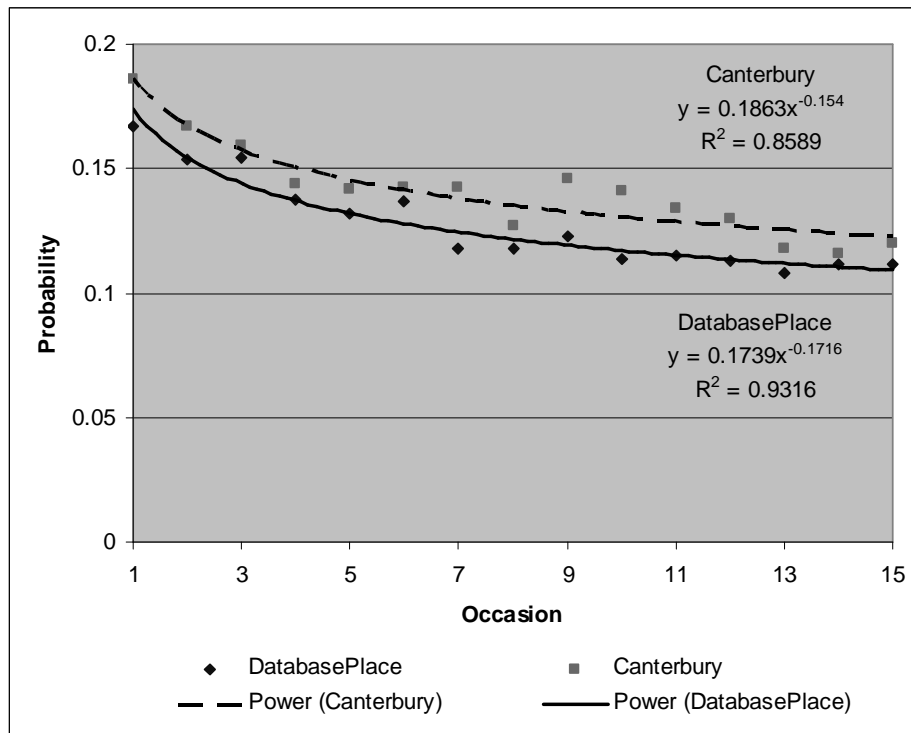


Figure 4. NORMIT learning curves for Canterbury and Database Place Students

Author Support

Building ITSs from scratch is a large, time-consuming process. To ease the development of further tutors we developed WETAS¹⁰, a web-enabled authoring shell that provides all the functionality necessary for constraint-based tutors, including student modeling, problem selection and feedback. WETAS can generate simple text-based interfaces automatically, whilst more complex interfaces are catered for by allowing the author to provide these resources in the form of custom HTML pages or Java applets. To develop a new constraint-based tutor using WETAS, the author provides the domain model and a set of problems with their solutions. LBITS and the web-enabled version of our database design tutor (EER-Tutor) were developed in WETAS, as well as several other constraint-based tutors, including Collect-UML described in the next section. WETAS can also be invoked via remote procedure calls, which allows tutoring functions to be embedded into existing applications. For example, the students at Canterbury University developed a UML diagrams tutor as a plugin for the CASE tool Borland Together.

WETAS dramatically decreases the time required to build a fully working tutoring system. Since its creation in 2001, WETAS has been used for five years by students in a graduate course on ITS at Canterbury University. The students are given just 3-4 weeks to develop a fully working tutor using WETAS in a fairly simple domain, such as English pluralization or adding fractions. In all cases, 89% or more of the students have created

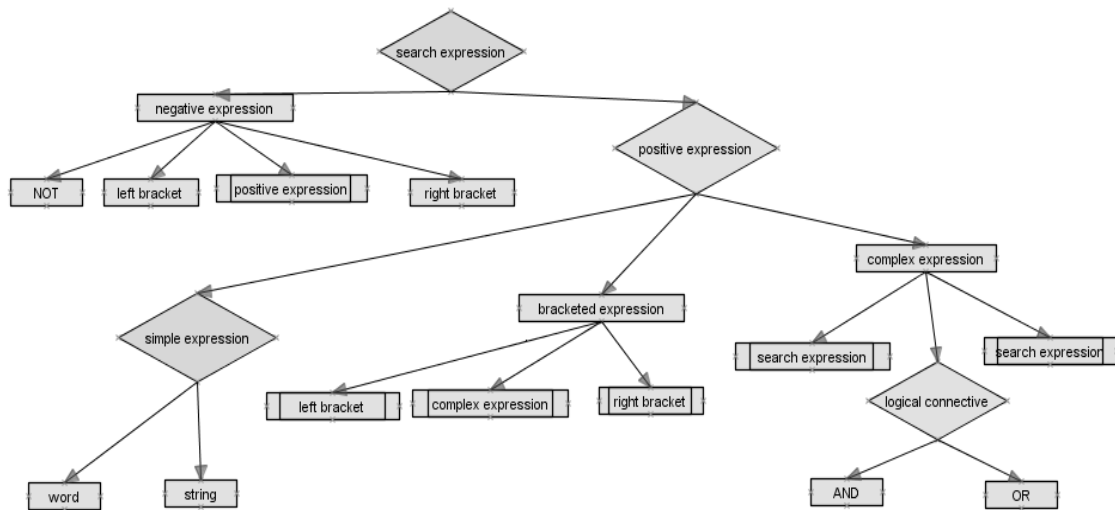


Figure 5. The ontology developed in WETAS-Ontology for a search language

fully functioning tutors in the time available. In 2006 a graduate student at Canterbury University developed a tutor for teaching German adjectives, which was used by a German class at the University. Two versions of the system were built for comparative testing of two interface styles, with the actual implementation taking around 40 hours to complete. The system received very positive reviews by both the students and their instructor, who is keen to develop more tutors to support the teaching of German. This illustrates how WETAS has made it feasible to rapidly develop and implement effective tutors in a classroom setting.

Although WETAS significantly shortens the development time, creating the domain model (the most difficult part⁴) is still hard. Domain models typically consist of hundreds of constraints: for example, the domain model of SQL-Tutor consists of almost 700 constraints, each taking over an hour to develop⁶. SQL-Tutor constraints were developed by an authoring expert; for novice authors the time per constraint would probably be larger. Novice authors experience two main hurdles: it is hard to work out what to model, and it takes skill to transform the conceptual model into actual constraints¹⁰. The latter problem is tackled to some degree in WETAS by using a custom constraint language that reduces constraints to pattern matches and logical connectors only. To overcome the second problem of working out what to model, we developed WETAS-Ontology, a system that lets the author create an ontology of the domain diagrammatically. WETAS-Ontology then generates the constraints based on the ontology. Figure 5 shows the ontology for a hypothetical search engine language created using the WETAS-Ontology tool.

The ontology is a combination of a taxonomy and a partonomy: each node represents a domain concept, where a diamond indicates that the sub-nodes are alternative specializations of this node (i.e. they have an “is-a” relationship with this node) and a rectangle indicates that any sub-nodes are components of this node and must appear in

the solution in strict order (i.e. they have a “part-of” relationship). For example, a logical connective can be either AND or OR; a complex expression consists of a search expression followed by a logical connective and another search expression. Rectangles with double edges represent concepts that have already been defined. Note that the latter could also have been achieved by allowing a node to have multiple parents. However, this would make the representation visibly more complex. This particular representation was chosen because it is simple to understand by novices and provides a clear visualization of the domain. Once the ontology has been created, WETAS-Ontology uses a set of templates to generate constraints directly from the nodes in the ontology. One constraint is created (per concept node) for each test we need to make, such as: Does this concept need to be used or not; Has the same instance of this concept been used by the student as the author; Has the student provided all required instances of this concept; Has the student used the correct sub-concepts?

```
(5 "Check whether you need one or more strings in your
answer."
; Relevance condition
  (MATCH IDEAL_SOLUTION (?* (^string ?IS_1) ?*))
; Satisfaction condition
  (MATCH STUDENT_SOLUTION (?* (^string ?SS_1) ?*)))

(16 "Are you sure you need complex expressions in your
answer?"
; Relevance condition
  (OR (MATCH STUDENT_SOLUTION (?* "AND" ?*))
      (MATCH STUDENT_SOLUTION (?* "OR" ?*)))
; Satisfaction condition
  (OR (MATCH IDEAL_SOLUTION (?* "AND" ?*))
      (MATCH IDEAL_SOLUTION (?* "OR" ?*))))
```

Figure 6. Generated constraint examples

Figure 6 shows two constraints generated from the ontology in Figure 5. Constraint 5 is for the concept “string” and checks whether or not a string is needed. The test for a string is more complex than can be easily represented in the ontology, so a macro “^string” has been used; such macros are hand written by the author in a language similar to the constraint language. Writing macros is an additional task to producing the ontology, however in practice few (if any) macros tend to be required. Constraint 16 checks whether or not the concept “complex expression” is needed in the student’s solution. In this case there is no easy way to test for this concept because it consists only of two alternative sub-concepts and no literal components. The generator therefore descends the tree until it finds sub-concepts with literal components (in this case “AND” and “OR”) and creates tests for each alternative sub-component.

WETAS-Ontology was trialed at the Adaptive Hypermedia 2006 Summer School on e-learning, held at the National College of Ireland in Dublin. Students were first asked to develop an ontology for the search language domain just described, and were given just

one hour to complete it, including around fifteen minutes spent instructing them in the use of the tools. At any time they could get the tool to create a set of constraints from their ontology, which was then automatically loaded into WETAS so they could test it (a problem/solution set was provided for them). Of the twelve students who participated in the exercise, half of them succeeded in creating an ontology that produced a fully functioning tutor in the time allocated, although all but one of these could have benefited from some refinement. Of the other six students, three had nearly completed the ontology. The remaining three had made some progress.

The students were also required to complete a group project in some area of adaptive web-based education systems, with six projects being offered. Two groups (representing around half of the total class) chose to use WETAS for their project, with one group using it to develop a tutor for English pluralization. They succeeded in creating a complete working tutor from scratch in just six hours, including developing a problem/solution set and building the domain model. For the latter they used WETAS-Ontology to create the set of constraints; they then wrote some low-level pattern matches to make the constraints more flexible and edited the generated feedback messages. The resulting tutor was of sufficient quality that it could be deployed in an elementary school.

WETAS-Ontology makes it easier to create the domain model, and allows the author to visualize the concepts of the domain via the ontology diagram tool. However, it only enables simple semantic checking. In particular, it is difficult to model multiple, highly dissimilar ways of satisfying a particular concept; the author must still code such constraints by hand. WETAS is therefore a useful tool for developers, but is not suitable for novice authors such as educators. ASPIRE¹¹ is an authoring environment currently under development that aims to be usable by teachers. Like WETAS-Ontology, ASPIRE provides rich domain-authoring support in addition to deploying ITSs. Authoring in ASPIRE starts with the author describing general features of the domain, such as whether the task is procedural or not, and specifying steps for procedural tasks. The author then describes the domain in terms of an ontology. Constraints that check for syntactic correctness can be generated directly from the ontology. The author is then asked to describe the structure of solutions in terms of the ontology, and provide a set of problems and their solutions. Finally, ASPIRE generates a set of constraints for checking the semantics of the answer using a machine-learning technique: alternative (correct) solutions are compared, and, if necessary, constraints are specialized or generalized to be consistent across all the given solutions. It is this last step that enables the system to model alternative correct solution approaches.

ASPIRE also provides the essential components of a tutoring system similar to WETAS, enabling the domain models generated using ASPIRE to be served as web-based tutors. In addition to running a collection of tutoring systems in parallel, ASPIRE also provides the functionality for managing teachers and students and student cohorts, which allows teachers to be able to assign ITSs for their students. We are currently evaluating ASPIRE with two novice authors, who are developing ITSs for very different domains (thermodynamics and accounting), as well as exploring its efficacy in the more complex

domains of computer program design (via Nassi-Shneiderman diagrams) and engineering mechanics.

Representing and Supporting Meta-Cognitive Skills

ITSs use their domain models and adaptive support to help students acquire declarative knowledge and problem-solving skills in the chosen domain. However, very successful students also understand how to learn effectively, and how to monitor their own cognitive processes. We have enhanced some of our constraint-based tutors to provide support for such higher-order, or “meta-cognitive”, skills. For example, NORMIT-SE and KERMIT-SE are versions of our data normalization and database design tutors respectively that have been extended to provide support for self-explanation, by asking students to explain their actions. Students are required to justify their problem-solving decisions when they make errors and thus practice self-explanation skills. The student is asked to explain an incorrect action in a series of questions, thereby leading him/her towards the correct solution through their own reasoning. Students are free to stop the dialogue at any point once they understand the problem and have determined how to recover from it.

Collect-UML is a constraint-based tutor that supports groups of students collaborating on a software design task. As well as teaching the students problem-solving skills in the area of UML class diagrams, it also strengthens their collaborative skills. Problem solving consists of three phases: students begin by negotiating the collaboration process and working on individual solutions. This is followed by a collaboration phase during which they develop a group solution. The latter phase involves communication between students via a chat tool. As well as providing feedback on the individual and group solutions Collect-UML also gives advice that aids them in their collaboration.

To be able to support these meta-cognitive skills, constraints need to represent more than just domain knowledge. We are now extending the use of constraint-based modeling to represent meta-cognitive knowledge as well. Two current projects include modeling explanation and communication skills as constraints. Collect-UML includes a model of collaboration skills represented in such a manner. An evaluation study performed in May 2006 showed that such support for collaboration results in improved declarative knowledge of good collaborative skills, as well as improving collaboration within groups. In another research project we are developing a meta-model of explanations skills that controls the self-explanation process and makes it more adaptive.

Conclusions

Our research has demonstrated that CBM is a very effective modeling approach that provides good foundations for successful instruction. Our constraint-based tutors have been thoroughly evaluated and proven to achieve significant learning gains. The interfaces of our systems are easy to use and reduce the working memory load by providing domain-specific information, visualizing the solution structure and structuring students’ thinking. Our tutors enforce good practices in the chosen instructional domain, and provide learning environments that are close to the real-world environment.

Evaluation studies have also shown that the wording of feedback is important; when feedback is worded according to the theory of learning from performance errors² students learn more. Furthermore, we have seen that asking students to provide reasons behind their actions results in learning declarative knowledge more efficiently. Recent achievements in advancing the capabilities of CBM include the use of constraints to represent not only domain-level knowledge, but also meta-cognitive knowledge. We have used constraints in two projects to represent the ideal models of collaboration and self-explanation skills.

We have also made significant advances in supporting the authoring process. WETAS, an authoring shell for developers, provides all the domain-independent tutoring functions, thus freeing up the author to perfect the all-important domain model. ASPIRE goes one step further and provides support for novice authors by automating many of the authoring tasks and providing comprehensive support for the others. Work continues on enhancements of ASPIRE and further development of our philosophy of authoring constraint-based tutors from domain ontology, including investigating the features of instructional domains that make them particularly suitable for CBM. ASPIRE will be a solid test-bed for this research. Furthermore, ASPIRE will be freely available via the Web from mid 2007 to anyone interested in building constraint-based tutors, and the ITSs developed in it will also be free to use; other researchers will therefore be able to confirm the effectiveness of our approach by developing their own constraint-based tutors and deploying them in ASPIRE. As the approach gains popularity, ASPIRE has the potential to deliver ITS to the world's classrooms.

Acknowledgments

This research was supported by grants U6430 and U6532 from the University of Canterbury, and eCDF grants 502 and 592. We are grateful for the contributions of Stellan Ohlsson and all present and past members of ICTG.

References

1. Anderson, J.R. et al. Cognitive Tutors: Lessons Learned. *Journal of the Learning Sciences* 4(2), 1995, 167-207.
2. Ohlsson, S. (1996). Learning from performance errors. *Psychological Review*, 103, 1996, 241-262.
3. Ohlsson, S. Constraint-based student modeling. *Journal of Artificial Intelligence and Education*, 3(4), 1992, 429-447.
4. Murray, T. An Overview of Intelligent Tutoring System Authoring Tools: Updated analysis of the state of the art. *Authoring tools for advanced technology learning environments*. 2003, 491-545.
5. Mitrovic, A., Koedinger, K. and Martin, B. A Comparative Analysis of Cognitive Tutoring and Constraint-Based Modelling. P. Brusilovsky, A. Corbett, F. de Rosis (Eds.) *Proc. 9th Int. Conf. User Modeling*, Springer-Verlag, LNAI 2702, 2003, pp. 313-322.
6. Mitrovic, A. and Ohlsson, S., Evaluation of a Constraint-Based Tutor for a Database Language. *Int. J. Artificial Intelligence in Education*, 10(3-4), 1999, 238-256.
7. Suraweera, P., Mitrovic, A. An Intelligent Tutoring System for Entity Relationship Modelling. *Int. J. Artificial Intelligence in Education*, 14(3-4), 2004, 375-417.

8. VanLehn, K. et al. The Andes Physics Tutoring System: Lessons Learned. *Int. J. Artificial Intelligence in Education*, 15(3), 2005, 147-204.
9. Koedinger, K.R., Anderson, J.R., Hadley, W.H. and Mark, M.A. Intelligent tutoring goes to school in the big city. *Int. J. Artificial Intelligence in Education*, 8, 1997, 30-43.
10. Martin, B. and Mitrovic, A. Domain Modeling: Art or Science? In: U. Hoppe, F. Verdejo & J. Kay (ed) Proc. 11th Int. Conf. Artificial Intelligence in Education, IOS Press, 2003, pp. 183-190.
11. Mitrovic, A. et al. Authoring constraint-based tutors in ASPIRE. M. Ikeda, K. Ashley, and T.-W. Chan (Eds.) Proc. 8th Int. Conf. Intelligent Tutoring Systems, Springer, LNCS 4053, 2006, pp. 41-50.

Author Biographies



Dr. Antonija Mitrovic is an Associate Professor at the University of Canterbury, and director of the Intelligent Computer Tutoring Group. Her research interests are in Intelligent Tutoring Systems, databases and GIS. She holds a PhD in intelligent education systems from the University of Nis, Yugoslavia.



Dr. Brent Martin is a Senior Lecturer at the University of Canterbury and a researcher in the Intelligent Computer Tutoring Group. He performs research in intelligent education systems and machine learning. He holds a PhD in intelligent education systems from the University of Canterbury, New Zealand.



Pramuditha Suraweera is a Research Fellow at the University of Canterbury and a researcher in the Intelligent Computer Tutoring Group. His research interest is in intelligent education systems. He holds a MSc in intelligent education systems from the University of Canterbury, New Zealand.